Community mining using three closely joint techniques based on community mutual membership and refinement strategy

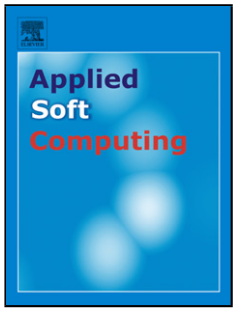
Shang, Ronghua; Liu, Huan; Jiao, Licheng; Ghalamzan Esfahani, Amir M.

DOI:
10.1016/j.asoc.2017.08.050

Document Version
Peer reviewed version

Citation for published version (Harvard):
Shang, R, Liu, H, Jiao, L & Ghalamzan Esfahani, AM 2017, 'Community mining using three closely joint techniques based on community mutual membership and refinement strategy', *Applied Soft Computing*, vol. 61, pp. 1060-1073. https://doi.org/10.1016/j.asoc.2017.08.050

Accepted Manuscript



Title: Community Mining Using Three Closely Joint Techniques Based on Community Mutual Membership and Refinement Strategy

Authors: Ronghua Shang, Huan Liu, Licheng Jiao, Amir M.Ghalamzan Esfahani

PII: S1568-4946(17)30532-X
DOI: http://dx.doi.org/10.1016/j.asoc.2017.08.050
Reference: ASOC 4440

To appear in: *Applied Soft Computing*

Received date: 3-10-2016
Revised date: 1-8-2017
Accepted date: 26-8-2017

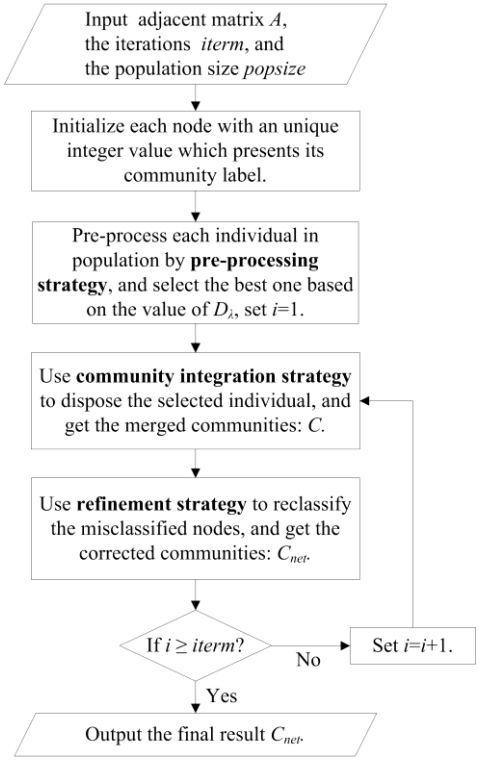
Please cite this article as: Ronghua Shang, Huan Liu, Licheng Jiao, Amir M.Ghalamzan Esfahani, Community Mining Using Three Closely Joint Techniques Based on Community Mutual Membership and Refinement Strategy, Applied Soft Computing Journalhttp://dx.doi.org/10.1016/j.asoc.2017.08.050

Community Mining Using Three Closely Joint Techniques Based on Community Mutual Membership and Refinement Strategy

Ronghua Shang[1], Huan Liu[1], Licheng Jiao[1], and Amir M. Ghalamzan Esfahani[2]

(1. Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, Xidian University, Xi'an, 710071, Shaanxi Province, China; 2. The Extreme Robotics Lab, University of Birmingham, UK.)

Graphical Abstract



Highlights

- We propose an algorithm to efficiently identify communities in a large network with a good balance between accuracy, stability and computation time.
- First, we propose an initial labeling algorithm, called ILPA, combining K-nearest neighbor (KNN) and label propagation algorithm (LPA).

- Next, we merge sub-communities using mutual membership of two communities.
- Finally, a refinement strategy is designed for modifying the label of the wrongly clustered nodes at boundaries.
- The experimental results on large-scale artificial networks and real networks illustrate the superiority of our algorithm.

***Abstract*:** Community structure has become one of the central studies of the topological structure of complex networks in the past decades. Although many advanced approaches have been proposed to identify community structure, those state-of-the-art methods still lack efficiency in terms of a balance between stability, accuracy and computation time. Here, we propose an algorithm with different stages, called TJA-net, to efficiently identify communities in a large network with a good balance between accuracy, stability and computation time. First, we propose an initial labeling algorithm, called ILPA, combining K-nearest neighbor (KNN) and label propagation algorithm (LPA). To produce a number of sub-communities automatically, ILPA iteratively labels a node in a network using the labels of its adjacent nodes and their index of closeness. Next, we merge sub-communities using the mutual membership of two communities. Finally, a refinement strategy is designed for modifying the label of the wrongly clustered nodes at boundaries. In our approach, we propose and use modularity density as the objective function rather than the commonly used modularity. This can deal with the issue of the resolution limit for different network structures enhancing the result precision. We present a series of experiments with artificial and real data set and compare the results obtained by our proposed algorithm with the ones obtained by the state-of-the-art algorithms, which shows the effectiveness of our proposed approach. The experimental results on large-scale artificial networks and real networks illustrate the superiority of our algorithm.

***Keywords*:** Community detection; K-nearest neighbor; community mutual membership; refinement strategy; large-scale complex networks.

1. Introduction

Complex networks can describe many real systems, such as biological protein systems [1], social networks [2], scientific collaboration networks [3] and the World Wide Web [4]. These systems usually contain a number of elements that may fully or partially share some common properties. In the abstract representation of these systems, a node stands for a specific individual element of a system where an edge between two nodes illustrates a certain structural relation. Hence, many real systems can be studied by the complex network model with different topological properties, such as the small world effect [5], the scale-free property [6] and the community structure [7, 8] to identify a structural pattern within the system, e.g. to safeguard the network or provide better service. Community detection or network clustering caught researchers' attention in the past years in the context of complex

networks [9-12]. This helped us to better understand social, biological and physical systems [12, 13]. In spite of many preeminent network-clustering algorithms [14-16], the definition of the community is not unique. Several popular definitions of a community from different perspectives are presented in the literature [17-19]. We consider communities that hold dense intra-connections and sparse interconnections, called modules or clusters, as per [17, 19, 20]. This qualitative description generates two concepts which are the strong community whose internal degree of each node exceeds its external degree and the weak community whose internal degree is larger than its external degree. Identifying such communities is significant for research. For example, scientists having the same research fields collaborate more frequently than others in different research fields [3] in a network of scientists and detecting these communities in the network provides us with information very useful for making decision or resource assignment.

We consider two categories of community detection algorithms [12]: (i) non-optimization based approaches that identify a set of nodes of the same community, e.g., the label propagation algorithm (LPA) [13], the network clustering algorithm based on the maps of random walks (Infomap) [15] and the fast hierarchical modularity optimization algorithm (BGLL) [16] and (ii) optimization based approaches that identify modules by forming an objective function, e.g. the divisive hierarchical clustering algorithm (GN) [9], the genetic algorithm for community detection (GA-net) [21], the hybrid genetic algorithm for community detection (Memetic-net) [22], the network clustering approach based on multi-objective GAs (MOGA-net) [23] and the multi-objective community detection algorithm (MOCD) [24]. In the second category, the selection of objective functions determines the accuracy of clustering to a certain extent [25, 26]. Hence, several qualitative definitions of scoring metrics have been proposed in community detection, such as modularity (Q) [9], community score (CS) [21] and some others [23, 27, 28]. Nevertheless, Q [9] has been widely used because of its simplicity and initial good results on GN benchmark networks [7] where it has been empirically shown to be efficient [29, 30].

Although the optimization based approaches have shown good results, different researchers have identified some of their shortcomings. First, Q just measures the difference between a real network and its randomized version. Complex networks with different hierarchical structures are difficult to be evaluated by a single objective function where its obtained results may be invalid. Next, maximizing the objective function Q is proven to be an NP-hard problem [31]. Last and most important, there is a limit of resolution [32, 33] for maximizing Q. For instance, clustering techniques based on modularity Q usually cannot find small-scale communities if the community size is small with respect to the total size of a network and the intra-communities have relatively sparse connections. Accordingly, it is hard to get satisfactory results for such networks with fuzzy community structure just by maximizing Q. In sum, the maximum Q doesn't always correspond to the best partitioning of a network.

In order to resolve this issue [32, 33], Li et al. [34] introduced another scoring function, called modularity density and denoted by D. The generic version, denoted by D_λ, contains a tunable parameter λ. Changing this parameter adapts the function to detect communities with different hierarchical structures, which can deal with the issue of the resolution limit. Network clustering techniques based on the modularity density have been studied extensively resulting in considerably satisfying results, e.g. a memetic algorithm combining the simulated

annealing and the tightness greedy algorithm (MA-SAT) [35], a network clustering approach with the clonal selection (CSA-net) [36] and a community detection algorithm based on an improved modularity density increment [37]. These algorithms are mostly based on evolutionary models like Memetic-net [22], GA-net [21] and MOGA-net [23]. To the best of our knowledge, evolutionary algorithms (EAs) are always considered being a kind of global optimization method. EAs neither suffer from optimization issues nor have limiting requirements like differentiability or continuity of objective functions. While some widely used clustering methods, such as k-means [38] and fuzzy c-means (FCM) [39], must be provided with the number of clusters in advance, network clustering based on EAs does not need this information because it can automatically determine the number of clusters through the decoding phase of the optimal individual. Consequently, these merits accelerate the widespread applications of EAs [27, 40-43]. Nevertheless, EAs get the optimal solution or the approximate solution of a specific issue only after evolving adequate generations. For a small-scale network, it will get a desired result in a reasonable period of time. However, for a large-scale network, this kind of generation-based optimization techniques cannot generate satisfying results within a limited time.

In order to resolve these deficiencies, we propose an algorithm, which is called TJA-net. Our proposed algorithm, similar to the methods based on EAs, does not need the number of clusters in advance, which needs a priori knowledge and it is hard to set. In fact, we often have no idea of the real number of clusters of many real networks; hence, automatically determining the number of final clusters is highly demanded [39]. Our proposed algorithm is composed of three closely jointed steps where each step tries to deal with a common challenging issue of network clustering. Comparing to EAs, our proposed algorithm only requires a small number of iterations to accurately identify the optimal partitions and obtain the optimal number of communities. Therefore, our algorithm shortens the search time to some extent. Finally, the proposed algorithm adopts the generic modular density D_λ as its objective function, which effectively alleviates the resolution limit caused by the optimization of Q. The algorithm is briefly reported as follows:

- First, we combine the K-nearest neighbor (KNN) [44, 45] and the label propagation algorithm (LPA). The combined algorithm is utilized for preliminary labeling of a network. This preprocessing process will take into consideration the classes of adjacent nodes and the degree of closeness between a node and its adjacent nodes. The algorithm updates the label of a node according to the closeness of the node to other nodes as well as according to the label of adjacent nodes. This method can accurately cluster some tightly connected nodes into a small-scale cluster, also called sub-community, in the initial stage very quickly.
- Second, the algorithm performs community integration strategy to obtain the correct communities/clusters from the sub-communities obtained by the first stage of TJA-net using a mutual membership function, which is used to measure the degree of membership between two sub-communities. The sub-communities whose closeness value is more than a given threshold will be iteratively merged into a bigger community resulting in an optimal number of communities that the algorithm obtains by modifying its solutions iteratively.
- Last, a refinement strategy is used to reclassify the nodes misclassified by the first two stages. In fact, the first step is only designed for achieving rapid clustering and we do not expect it to have a high clustering accuracy.

Hence, TJA-net misclassifies several nodes during the first stage. We modify the proposed node-to-community membership function $f_{MS\text{-}NC}$ [46] so that interference from anthropogenic factors is highly reduced. This ensures high accuracy of clustering results and greatly reduces the manual workload.

We designed and performed a series of experiments using both artificial and real network datasets to show the effectiveness and usefulness of our proposed algorithm. Moreover, we compare the results obtained by our approach and the state-of-the-art algorithms to validate our approach. The comparison shows that our proposed algorithm outperforms those state-of-the-art algorithms on large-scale networks in terms of the accuracy and computation time. In section 2, we present the formulation of our approach. A large number of experiments of our proposed algorithm are arranged in section 3 and we conclude in section 4.

2. The proposed algorithm

Let $G=(V, E)$ represent an unsigned network where V and E represent the aggregations of nodes and edges, respectively. In addition, $|V|=n$ is the number of nodes and $|E|=m$ is the number of edges in G. We call a subset of G a community $C_k \subset G$ (k=1, 2, …, l), where l is the number of communities. Moreover, A represents a priori knowledge of an input network in the form of the adjacent matrix. For example, A_{ij}=1 if there exists an edge between the i_{th} and the j_{th} node; otherwise, A_{ij}=0. Finally, $d_i = d_i^{int} + d_i^{ext}$ denotes the degree of node $i \in C_k$, where $d_i^{int} = \sum_{j \in C_k} A_{ij}$ and $d_i^{ext} = \sum_{j \notin C_k} A_{ij}$ are the internal degree and external degree of the i_{th} node, respectively.

2.1 Representation and Initialization

In this study, the encoding method of TJA-net will be introduced. First, we initialize a number of individuals where each individual is a representation of the network. To the best of our knowledge, there are two encoding patterns commonly used: (i) string_based encoding [47] and (ii) locus_based encoding [48]. We use the string_based encoding because its encoding pattern is easy to implement and it is effortless to decode.

A schematic of the encoding and decoding pattern is shown in **Fig.1**, which illustrates the main process of the proposed approach. This figure shows that we first assign an identifying number to every node, which indicates the community to which a node belongs, which we call '*community label*'. Ideally, nodes belonging to the same community are expected to have the same community label. The obtained solution, namely community labels, is decoded to get the network partition. The network in the bottom right of **Fig.1** shows the nodes after decoding with different shapes and colors. Those red circles show the nodes belonging to community 1 and the green hexagons show those nodes identified as community 2.

2.2 Preprocessing strategy

Before presenting the preprocessing phase of the TJA-net, we need to present some preliminaries. For example, K-nearest neighbor (KNN) [44, 45], which is proposed by Cover and Hart [44] in 1967, is one of the widely used and relatively simple machine learning algorithms. In KNN, a label is assigned to a node according to the labels

assigned to the nodes in its vicinity. However, the number of clusters, denoted by K, needs to be set a priori in this method. This highly affects the result of the clustering approach whereas setting the value of K either requires domain knowledge or needs trial and error. Next, Raghavan et al. [14] proposed an agglomerative method called label propagation algorithm (LPA) that doesn't require the information of the whole network and has nearly linear time complexity. Hence, it is well suited for the large-scale network clustering. LPA iteratively operates on a network and labels a node based on the labels assigned to nodes in its neighbor. Eventually, the assigned label to the node will stay changeless after a small number of iterations where the label will be in correspondence with the labels of the majority of its neighbor nodes. This algorithm converges very fast, e.g. after five iterations. LPA has shown good convergence property and therefore it has been used in different fields including community detection [30, 49].

In spite of the good algorithmic properties of LPA, it has some drawbacks: (i) in LPA only the labels of all adjacent nodes are considered while the degree of closeness between nodes is not used; (ii) LPA does not effectively assign a label to a node if the labels of the majority of nodes in its neighborhood are not identical. The second drawback happens if the network has lots of weak communities in which the true communities are not well separated. For such networks, LPA cannot correctly identify communities or even take the entire network as a community. We will discuss this issue in detail in section 3.5.

To resolve these shortcomings, we propose a variation of LPA called ILPA, which combines KNN and LPA. First, we define a cohesion index for any two adjacent nodes, which is the number of nodes in the neighborhood of the two adjacent nodes. Intuitively, two nodes with more common neighbors are more likely to be in the same community. For a set of given nodes $V=\{v_1,v_2,\ldots,v_n\}$ in a network, we define the degree of closeness between node v_i and v_j ($v_i, v_j \in V$) as follows:

$$S_{ij} = \left|\Gamma(v_i) \bigcap \Gamma(v_j)\right| + 1 \qquad (1)$$

where $\Gamma(v_i)$ denotes the set of adjacent nodes corresponding to node v_i, which have direct connections to v_i, including v_i itself. The constant indicates that when the two adjacent nodes don't have any common neighbor, we can still maintain $S_{ij}=1$ in order to indicate that there is an edge between v_i and v_j.

The preprocessing of ILPA is reported in **Algorithm 1**. At this stage, the real community label may be different than the one used to initialize the labeling at the beginning. The labeling of each node is effectively and iteratively assigned until the labeling converges to a final solution by ILPA. A key advantage of the proposed preprocessing is that a relatively stable solution is obtained after only a few 5 iterations in many of our experiments. In contrast to KNN that needs the value of K to be set in advance, in our approach the value of K is automatically associated with the number of adjacent nodes of the pending node by considering the nodes connected to the node of interest when we calculate their similarities. In this way, the labels of nodes adjacent to a node of interest only affect the label of that node and the value of K is merely specific to that node. We consider the value of K being half of the number of its adjacent nodes.

Algorithm 1. Procedures of the first stage: Preprocessing

Input: The adjacent matrix of a network: A, the number of nodes: n, the iterations: m_iter.

Output: Sub-communities of the input network: SC.

Step 1. Assign a unique number to each node, that is, $label(V)=\{1, 2, \ldots, n\}$.

Step 2. Set t=1.

Step 3. Set i=1.

Step 4. Calculate the closeness of node v_i to its adjacent nodes, then sort the closeness in descending order: $S_i=\{S_{i1}, S_{i2}, \ldots, S_{iq}\}$ (i=1, 2, …, n), q denotes the number of adjacent nodes of v_i.

Step 5. Set $K=\lfloor q/2 \rfloor+1$. If K is an even number, reset K=K-1; then go to **Step 6**.

Step 6. Get the first K values in S_i, take their corresponding nodes from adjacent nodes of v_i, and find most of the same labels (denoted as r) to which these K nodes correspond. Use label r to update the label of v_i.

Step 7. Update i=i+1 If $i \le n$, go to **Step 4;** otherwise, go to **Step 8**.

Step 8. Update t=t+1 If $t \le m\_iter$, go to **Step 3**; else, go to **Step 9**.

Step 9. Output the sub-communities: SC.

To illustrate the difference between our proposed approach, namely ILPA, and LPA, we depict it in **Fig. 2**. The topological graph is shown in **Fig. 2** (a) where the nodes $\{v_1, v_2, v_3\}$ belonging to community C1 are labeled with '1' whereas the nodes $\{v_5, v_6, v_7, v_8, v_9\}$ belonging to community C2 are labeled with '2'. In this figure, node v_4 is a pending node. As we process v_4 by LPA, its adjacent nodes, namely $\{v_1, v_2, v_3, v_5, v_7, v_8\}$, are taken into consideration. Node v_4 has three edges connecting it with nodes belonging to C1 and three edges connecting it with nodes belonging to C2. Hence, no valid class can be assigned to v_4 by LPA and the label updating of v_4 entirely depends on a random decision. In contrast, our approach first records the closeness values of v_4 to its neighbors as shown in **Table 1**. Next, the algorithm sets K=3, which is the half of the number of nodes connected to v_4. Using eq. (1), we obtain a set of nodes $\{v_1, v_2, v_3\}$ which are closest ones to v_4. Finally, ILPA obtains label '1' for v_4 because the set of closest nodes belongs to community C1, which is shown in **Fig. 2** (b).

This example shows that ILPA, which is a combination of LPA and KNN, is more efficient and it is less sensitive to the noise. In addition, we empirically noticed that there is no difference between ILPA and LPA if the value of K in ILPA is set to the number of adjacent nodes. It is also obvious that changing the value of K can change the final labeling of the pending node, to some extent, since it determines a set of nodes adjacent to the pending node. **Algorithm 1** reports in detail the procedures of this preprocessing strategy.

2.3 Community integration strategy

Through the preprocessing stage, a number of small-scale clusters, called sub-communities, are identified because they may be tightly connected. This can be guaranteed with the minimum effort by controlling the value of K. The smaller value of K indicates that larger number of sub-communities will be potentially obtained. As a community is usually composed of several sub-communities, we need to merge those available sub-communities to obtain a reasonable number of network segments. We call this strategy "*community integration*".

Angelini et al. [50] presented a definition of closeness to measure the similarity of two local communities. We were inspired by this approach and designed a community membership function f_{CM}. Consider a given network

$G=(V, E)$ and a pre-division denoted by $SC(G)=\{SC_1, SC_2, \ldots, SC_w\}$, $w \geq l$, SC_k ($k=1, 2, \ldots, w$) is a sub-community in the network where w is the number of sub-communities. We take $\Gamma(SC_k)$ afterwards as a set of nodes adjacent to sub-community SC_k. This is the aggregation of the external nodes that connect to SC_k as follows:

$$f_{CM}(SC_i, SC_j) = \frac{\left|\Gamma(SC_i) \bigcap SC_j\right|}{\left|\Gamma(SC_i)\right|} \quad (2)$$

where $\Gamma(SC_i) \cap SC_j$ represents the connectivity degree between sub-community SC_i and SC_j, and $|\Gamma(SC_i)|$ represents the number of adjacent nodes that connect to SC_i. Hence, $f_{CM}(SC_i, SC_j)$ evaluates the degree of closeness of SC_i and SC_j. Given a pre-division $SC(G)=\{SC_1, SC_2, \ldots, SC_w\}$, we define a function denoted by f_{CI} for calculating the degree of mutual membership between two communities, as follows:

$$f_{CI}(SC_i, SC_j) = f_{CM}(SC_i, SC_j) + f_{CM}(SC_j, SC_i), \ (i, j = 1, 2, \ldots, w) \quad (3)$$

We call this function mutual membership function. The value of mutual membership of any two communities is calculated by considering a threshold δ for the value of f_{CI}. We merge the sub-communities according to the algorithm reported in **Algorithm 2**. It should be mentioned that, merging any two sub-communities requires two conditions: (i) the membership value of sub-communities is greater than the threshold δ, and (ii) the merged sub-communities result in an increased modularity density. We name this process as community integration strategy (CIS).

Algorithm 2. Procedures of the second stage: Community integration strategy

Input: The sub-communities of a network: $SC(G)=\{SC_1, SC_2, \ldots, SC_w\}$, the merging threshold: δ.

Output: The detecting result after merging: $C(G)=\{C_1, C_2, \ldots, C_l\}$, $l \leq w$.

Step 1. Set $i=1$.

Step 2. Set $j=1$.

Step 3. Get two sub-communities SC_i and SC_j. Then, calculate the mutual membership value between SC_i and SC_j, denoted by $f_{CI}(SC_i, SC_j)$.

Step 4. If $f_{CI}(SC_i, SC_j) \geq \delta$, sub-community SC_i and SC_j are merged, then calculate the value of D_λ. If D_λ declines, reset the individual to the last status.

Step 5. $j = j+1$. If $j > w$, go to **Step 6**. Otherwise, go to **Step 3**.

Step 6. $i = i+1$. If $i > w$, go to **Step 7**. Otherwise, go to **Step 2**.

Step 7. Output the merging communities: $C(G)=\{C_1, C_2, \ldots, C_l\}$.

2.4 Refinement strategy

Using the first two steps of our approach, which are mentioned before, a good partitioning can be obtained. However, there may be some misclassifications due to the instability of these steps. Here, we only stress the misclassification of the boundary nodes that have edges with more than one community. Since the community integration strategy cannot implement local search, we define a refinement strategy (RS) based on the approach proposed in [46] to resolve this issue. In this work, an evaluation function referred to as $f_{MS\text{-}NC}$ is proposed to modify the misclassified nodes, which considers both the possibility that a node belongs to a community and the capacity that a community accepts a node. The $f_{MS\text{-}NC}$ can be written as follows:

$$f_{CI}(SC_i, SC_j) = f_{CM}(SC_i, SC_j) + f_{CM}(SC_j, SC_i) \quad (4)$$

where, α, β are two adjustable parameters, controlling the proportion of the capacity of the node selection to the capacity of the community selection. For the sake of simplicity, we use i instead of v_i. Hence, d_i is the degree of node i, $|c|$ represents the number of nodes in community c and $J_{i,c}$ is the number of edges between node i and community c, i.e., $J_{i,c} = \sum_{j \in c} A_{ij} (i \notin c)$. The concept of the community integration strategy is as follows: for a boundary node i and for the communities in its neighborhood, counting the values of $f_{MS\text{-}NC}$, we select the maximum value of $f_{MS\text{-}NC}$, denoted by $max\_f_{i,c}$. Finally, node i will be reclassified as community c corresponding to the $max\_f_{i,c}$. This process is formulated in eq. (5) where $label(i)$ represents the label of node i.

$$label(i) = label(\arg\max_{c} \{ f_{MS-NC}(i,c), c \in \{C_1, C_2, \ldots C_l\}\}) \quad (5)$$

There are two adjustable parameters α, β in eq. (5) that need to be adjusted to obtain a good labeling. However, the values of these parameters have little effect on the clustering, and their tuning process requires a lot of effort. To address these defects, we redesign a node-to-community membership function, denoted by f_{INC}, which is formulated in eq. (6). This better captures the relation between nodes and communities without having parameters to be tuned.

$$f_{INC} = \frac{1}{2}\left(\frac{J_{i,c}}{d_i} + \frac{J_{i,c}}{d_c^{ext}} \right), (i = 1, 2, \ldots, n) \quad (6)$$

where $d_c^{ext} = \sum_{i \in c, j \notin c} A_{ij}$ indicates the connection degree of community c with other communities and $0 \le J_{i,c}/d_i \le 1$, $0 \le J_{i,c}/d_c^{ext} \le 1$; thus, $f_{INC} \in [0, 1]$. Considering that the number of nodes does not affect the ability of a community to accept other nodes, we use the external degree of a community instead of the size of the community. There are fewer misclassified nodes after using the first two procedures if a network has a strong structure. By strong structure, we mean that the internal connections between the nodes of a community are denser than the external connections with of the nodes with other communities. In a network, RS can play an important role only if there exist overlapping nodes, like the node v_4 in **Fig.2**. Intuitively, the probability that an external node belongs to a community is very high if the majority of boundary nodes in the community have connections with the same external node. All boundary nodes are arranged by the refinement strategy and only those nodes that satisfy some conditions will be reclassified and assigned to a more appropriate community reported in **Algorithm 3**.

Algorithm 3. Procedures of the third stage: Refinement strategy

Input: The adjacent matrix of an unsigned network: A, Communities after merging: $C(G)=\{C_1, C_2, \ldots, C_l\}$, l denotes the number of communities.

Output: The optimal solution of network clustering: C_{net}.

Step 1. Find all boundary nodes, namely a node has connections with multi-communities, denoted by $edge\_node = \{v_{b_1}, v_{b_2}, \ldots, v_{b_p}\}$, where p is the number of boundary nodes.

Step 2. Set i=1.

Step 3. Get a boundary node v_{b_i}, and record its current label $label(v_{b_i})$. Find all neighbor communities of v_{b_i}, denoted by $C_{neighbor}=\{C_{i1}, C_{i2}, \ldots, C_{it}\}$, where t is the number of the neighbor communities.

Step 4. Calculate the values of f_{INC} between node v_{b_i} and each of the neighbor communities $C_{neighbor}$, denoted by

$value\_f_{INC} = \{f_{C_{i1}}, f_{C_{i2}}, \ldots, f_{C_{it}}\}$.

Step 5. Find the maximum value in *value_fINC*, denoted by $f_{C_{ij}}, j \in \{1,2,\ldots,t\}$.

Step 6. Assign node v_{b_i} to community C_{ij}. Then calculate the value of D_λ. If D_λ increases, go to **step 7**. Otherwise, v_{b_i} should be reassigned to its original community.

Step 7. $i=i+1$. If $i > p$, go to **Step 8**; Otherwise, go to **Step 3**.

Step 8. Output the optimal solution: C_{net}.

2.5 Modularity density D_λ

Li et al. [34] presented the modularity density, denoted by D_λ, for dealing with the resolution limit with an adjustable parameter λ. The value of λ determines the network structure at different resolutions, which may result in satisfactory output. For an unsigned network $G=(V, E)$, a clustering result of network G can be denoted by $C(G)=\{C_1, C_2, \ldots, C_l\}(l=1, 2, \ldots, n)$, where C_i represents the set of nodes belonging to the i_{th} community. Let $d_{C_k}^{int} = \sum_{i,j \in C_k} A_{ij}$ and $d_{C_k}^{ext} = \sum_{i \in C_k, j \in \bar{C}_k} A_{ij}$ where $d_{C_k}^{int}$ is the internal degree of community C_k and $d_{C_k}^{ext}$ gives the external degree of community C_k. Hence, the modularity density D_λ is formulated in eq. (7), as follows:

$$D_\lambda = \sum_{i=1}^{l} \frac{2\lambda d_{C_i}^{int} - 2(1-\lambda) d_{C_i}^{ext}}{|C_i|} \tag{7}$$

where $|C_i|$ is the number of the nodes in community C_i and $\lambda \in [0,1]$. Under some conditions, D_λ can be regarded as a combination of the ratio association [51] and the ratio cut [52]. If $\lambda=0$, eq. (7) turns into ratio cut; otherwise eq. (7) is the ratio association. Furthermore, eq. (7) is the expression of the modularity density D if $\lambda=0.5$. In general, optimizing the ratio association allows us to obtain smaller communities while optimizing the ratio cut tends to partition networks into large communities. Hence, by changing the parameter λ, our proposed approach efficiently deals with the networks with different resolutions giving satisfactory clustering results.

2.6 TJA-net framework

Here, we can form our approach, called TJA-net, consisting of **Algorithm 1**, **2** and **3** that were discussed in the previous sections. A pseudo code of our proposed algorithm is presented in **Algorithm 4**. In the next sections, we will present a series of experiments to validate the superiority of TJA-net.

Algorithm 4. The whole framework of TJA-net

Input: Population size: *popsize*, the number of iterations: *iterm*, the adjacent matrix of an unsigned network: *A*.

Output: The detected communities by TJA-net.

Step 1. Initialize each node with a unique integer value, which presents its community label.

Step 2. Pre-process each individual in the population by **Algorithm 1**. Then, select the best individual in the population based on the obtained value of D_λ.

Step 3. Set $i=1$.

Step 4. Use **Algorithm 2** to dispose the selected individuals; then, get the merged communities: *C*.

Step 5. Use **Algorithm 3** to amend the misclassified nodes, and get the corrected communities: C_{net}.

Step 6. $i=i+1$. If $i>iterm$, terminate the algorithm and go to **Step 7**; otherwise, go to **Step 4**.

Step 7. Output the detected communities.

3. Experiments

In this section, we present a series of experiments to demonstrate the good performance of our proposed approach. In addition, we will compare the results obtained by our approach and some state-of-the-art approaches demonstrating the superiority and efficiency of our approach. We use *robustness*, *precision* and *running time* of the algorithms for comparing the results of these approaches. We implemented all algorithms in C++ programming language running on a machine with the processor Intel(R) Core(TM) i5-2450M CPU @ 2.50GHz, the memory is 6.00 GB and the operating system is Windows 7.

3.1 Evaluation index

First, we present an evaluation index called normalized mutual information (*NMI*) introduced in [53] that is often applied to evaluate the similarity between the detected communities and the true communities. Given two partitions *A* and *B* representing the true and the detected communities, respectively, *NMI* is defined in eq. (8) as follows:

$$I(A,B)=\frac{-2\sum_{i=1}^{N_A}\sum_{j=1}^{N_B}H_{ij}\times\log(H_{ij}\times n/H_{i\cdot}H_{\cdot j})}{\sum_{i=1}^{N_A}H_{i\cdot}\times\log(H_{i\cdot}/n)+\sum_{j=1}^{N_B}H_{\cdot j}\times\log(H_{\cdot j}/n)} \tag{8}$$

where N_A and N_B represent the number of communities belonging to *A* and *B*. H_{ij} is an element of the matrix *H*, and represents the number of nodes that the community *i* in partition *A* and the community *j* in partition *B* have in common where *H* is a confusion matrix. $H_{i\cdot}$ ($H_{\cdot j}$) represents the sum of elements in the *i*-th row (the *j*-th column) of the matrix *H*. Besides, *I*(*A*,*B*) belongs to [0, 1] where *I*(*A*,*B*) with a higher value indicates a better clustering result. If *I*(*A*,*B*)=0, the partition *A* and *B* are completely different whereas *I*(*A*,*B*)=1 shows that the partition *A* and *B* are completely identical. However, the *NMI* index can only be applied to evaluate a network whose true partition is known. Hence, we take the modularity density *D* as another evaluation index for a network whose true partition is unknown where λ=0.5 in D_λ.

3.2 Compared algorithms

To compare the results obtained by the proposed algorithm, we use Memetic-net [22] and CSA-net [36], which work based on the modularity density D_λ. In addition, we use other four optimization-based approaches, namely BGLL [16], Infomap [15], MODPSO [28] and NM_CI-net [46], respectively.

Memetic-net [22] is a hybrid evolutionary algorithm (hybrid EA) which combines the hill-climbing method and the genetic algorithm (GA). With regard to CSA-net, it [36] is established on the ideology of artificial immune system and it is an immune clone algorithm using an antibody population initialization mechanism and a hypermutation strategy. Both Memetic-net and CSA-net utilize D_λ as their objective function. These approaches can identify communities at different resolutions through adjusting the parameter λ. Blondel et al. [16] proposed BGLL that takes a local community integration strategy to optimize *Q* by combining a greedy strategy for iterative optimization. It merges two communities only if the resulting local modularity increases. Rosvall and Bergstrom [15] proposed Infomap using the probability flow of random walks on a network as a proxy for information flows

in the real system. This approach decomposes the network into clusters through compressing a description of the probability flow. Gong et al. [28] proposed a multi-objective discrete particle swarm algorithm (MODPSO). The label propagation (LP) was introduced to initialize its population. The authors also designed a perturbation operator to better guide the direction of particles. Furthermore, they took the kernel K-means and the ratio cut as their objective functions for managing unsigned networks. In order to verify the validity of the preprocessing strategy, we also take a variant of TJA-net as a comparison algorithm by adopting LPA as the preprocessing mechanism. We call this variant TJA_v-net.

3.3 Testing networks

We use two classical testing networks for experiments: (i) computer-generated networks and (ii) real networks.

Computer-generated networks: Lancichinetti et al. [54] proposed GN extended benchmark networks based on the classical GN benchmark networks [7]. This kind of network possesses 128 nodes and can be divided into four communities with the same size. It contains a mixing parameter γ that controls the connection degree between communities where a smaller value of γ indicates the network has a stronger community structure, which is easier to be detected. In contrast, a higher value of γ will result in more blurred boundaries, which makes detecting a harder task. The value of γ is considered being {0.0, 0.05, 0.1, 0.15, …, 0.5} giving us 11 computer-generated networks. However, both the average degree of each node and the size of each community in the GN extended networks are the same, which may not be realistic in real-world networks. In order to make an authentic simulation of real-world systems, Lancichinetti et al. [55] proposed LFR benchmark networks, which require several tuning parameters for controlling the structure of networks, e.g. the network scale, the degree distribution of nodes, the community size and the mixing parameter γ. In this work, we employ three types of the LFR networks with the scale of 1000 (LFR1), 5000 (LFR2), and 10000 (LFR3) nodes with other corresponding parameters as follows:

(1) LFR1: the average degree of each node: $k_{average}$=20, the maximum degree of nodes: k_{max}=50, the degree distribution exponents: τ_1=2, τ_2=1, the community size: [C_{min}, C_{max}]=[10,50];

(2) LFR2: $k_{average}$=20, k_{max}=50, τ_1=2, τ_2=1 and [C_{min}, C_{max}]=[20,100];

(3) LFR3: $k_{average}$=20, k_{max}=50, τ_1=2, τ_2=1 and [C_{min}, C_{max}]=[20,100].

In our experiments, we consider the parameter γ in LFR1 being {0.0, 0.05, 0.01, ..., 0.70} giving us 15 networks. Similarly, we consider γ in LFR2 and LFR3 being {0.0, 0.05, 0.01, ..., 0.90} giving us 19 networks.

Real networks: we also use six types of real-world networks for comparing our approach with the state-of-the-art approaches. The ground truths of three networks are known and the other three are unknown, i.e. the Zachary's karate network [56], the dolphin social network [57], the American college football network [7], the net-science network [58], the power grid network [5] and the PGP network [59]. The corresponding data of these networks are reported in **Table 2**.

3.4 Experimental results and analysis

To show the capabilities and superiority of our approach, we need to build and use different networks with different scales and properties. First, we present the computer-generated networks.

3.4.1 Experimental results on computer-generated networks

(1) The results obtained on GN extended benchmark networks: we use CSA-net and Memetic-net with the parameters reported in their original papers as the comparison algorithms in this experiment for validation. We also use the same parameters for two proposed algorithms, namely TJA_v-net and TJA-net, as follows: the population size, *popsize*=20; the threshold of community integration, δ=1; the number of iterations, *iterm*=5. The parameter λ in the objective function D_λ is considered being {0.2, 0.3, 0.4, ..., 1.0}. The maximum *NMI* values over 30 runs on these 11 networks are shown in **Fig.3**.

While the performance of TJA-net and Memetic-net are identical in the case $0.2 \leq \lambda \leq 0.4$ and for all γ, CSA-net only outperforms others for $0.2 \leq \lambda \leq 0.4$ and $\gamma = 0.4$. In addition, TJA-v-net performs worse than others only for $\lambda \leq 0.6$. On the other hand, both TJA-net and TJA_v-net significantly outperform CSA-net and Memetic-net in accuracy for $0.8 \leq \lambda \leq 1.0$. These results also show that even though the communities in some cases are easily identifiable, namely for $0.2 \leq \gamma \leq 0.3$ in the first two subfigures in **Fig.3**, TJA_v-net is not effective i.e. *NMI*=0.0. Nonetheless, this pattern changes for $\lambda > 0.7$ i.e. TJA_v-net outperforms Memetic-net and CSA-net for these networks. In general, TJA-net can be selected as the best among these four algorithms in terms of stability and it has a good detecting accuracy.

Due to the excessive amount of data, we cannot present all the experimental data here and we only report part of the data in **Table 3**.

The testing networks are GN extend benchmark networks at λ=0.3, 0.6 and 0.9 where $0.1 \leq \gamma \leq 0.5$. In the table, boldface numbers represent the best result in terms of the *NMI* index obtained by TJA-net. It can be observed in the table that TJA-net performs slightly worse than its variant TJA_v-net for λ=0.9 and $\gamma \leq 0.3$. Nonetheless, it performs much better than the other two algorithms.

(2) The results obtained on LFR benchmark networks: in this section, we find the communities in the networks of LFR1, LFR2 and LFR3 by using BGLL, Infomap, NM_CI-net, MODPSO, TJA-net and its variant, namely TJA-v-net. Because Memetic-net and CSA-net have not been designed for large-scale networks and they perform poorly on these networks, we do not report their corresponding results in this section. All the algorithms ran 10 times on the networks to obtain the maximum *NMI* values, which are shown in **Fig.4** (a), (b), (c). Because there are no overlapping nodes in all three kinds of computer-generated networks, the refinement strategy has little impact on the clustering results. Therefore, we do not use the refinement strategy and we only use the first two steps of our proposed approach. In addition, since the objective function of TJA-net and TJA-v-net contains an adjustable parameter λ, we need to fix it here, which is set to be 0.5.

If $0.0 \leq \gamma \leq 0.6$, only TJA-net, Infomap, and MODPSO are able to identify the correct structure, i.e. *NMI*=1.0, on

LFR1, as shown in **Fig.4** (a). However, Infomap cannot identify the correct structure of LFR1 for $\gamma > 0.6$ resulting in *NMI*=0.0. On the other hand, while TJA-net can detect the correct communities for $\gamma \leq 0.6$ and its performance is amongst the best ones for $\gamma \leq 0.65$, it only performs slightly worse than NM-CI-net for $\gamma = 0.7$. This indicates that TJA-net performs very well and robustly for the networks with fuzzy community structure. In addition, we observe that MODPSO can also identify the correct structure of LFR1 for $0.0 \leq \gamma \leq 0.6$ similar to TJA-net. The values of *NMI* obtained by MODPSO and TJA-net are bigger than the ones obtained by other approaches; however, the performance of MODPSO is inferior to our proposed TJA-net for $\gamma \geq 0.65$. On the other hand, NM_CI-net cannot correctly detect communities even though the communities in these networks are easily identified for $0.10 \leq \gamma \leq 0.50$.

The maximum *NMI* values by all the algorithms on LFR2 are shown in **Fig.4** (b). The experimental results on LFR2 in **Fig.4** (b) show that TJA-net, TJA_v-net, Infomap and MODPSO can fully detect the network partitions for $0.0 \leq \gamma \leq 0.5$. Although the detecting accuracy of TJA-net falls monotonically from 1.0 to less than 0.6 and it performs worse than MODPSO, Infomap and TJA_v-net for $0.55 \leq \gamma \leq 0.75$, TJA-net outperforms other approaches except for MODPSO and $\gamma \leq 0.8$. Furthermore, we observe that TJA_v-net loses its effectiveness on the networks for $\gamma \geq 0.7$ resulting in $NMI = 0.0$. The detecting accuracy obtained by Infomap decreases rapidly for $\gamma > 0.75$ where at $\gamma = 0.80$ it becomes zero. In sum, only MODPSO and TJA-net can maintain stability and a high detecting accuracy for $0.75 < \gamma < 0.90$ where they result in $NMI \geq 0.5$. This figure also shows that only MODPSO slightly outperforms our proposed approach on the LFR2 networks.

Fig. 4 (c) shows the best *NMI* values obtained by different approaches on LFR3. The detecting accuracy obtained by TJA-net improves significantly for $\gamma > 0.80$ such that it outperforms MODPSO. This is in contrast to the results on LFR2 shown in **Fig.4** (c). Although MODPSO obtains complete correct partitioning at $\gamma = 0.00$, i.e. $NMI = 1.0$, it cannot achieve correct partitioning on the rest networks where their corresponding *NMI* values are very close to 1.0. Again, we only report part of the results obtained on LFR3 in **Table 4**. These results show that TJA-net outperforms others in the most cases and its performance is near optimal for the rest. Moreover, TJA-net performs well in terms of stability even though the network is hard to cluster, e.g. for a network with $0.50 \leq \gamma \leq 0.90$.

Table 4 shows that the trends of changes of the maximum *NMI* by BGLL and NM_CI-net are very similar. Although TJA_v-net does perform less effectively than Infomap, these two have similar results. For example, they both obtain correct detecting results at high resolutions where their performance qualities decline sharply at low resolutions where they eventually drop to zero. Although TJA-net and MODPSO adopt two completely different algorithm frameworks for network clustering, they perform most efficiently in terms of stability and accuracy. Both algorithms have achieved a good tradeoff between accuracy and robustness requirements. It is difficult to distinguish which one is better because both two obtain remarkable network clustering results.

3.4.2 Experimental results on real networks

We now examine TJA-net on real-world networks. First, we compare the detecting results at different resolutions, on three real networks, denoted by N1, N2, and N3, with known ground truths to verify the efficiency of our

approach. Here, we adopt four algorithms, namely Memetic-net, CSA-net, TJA-net and TJA_v-net for comparison where their corresponding average *NMI* values are shown in **Fig. 5** over 30 times running. This figure shows that there exists a huge difference between the detecting results at different resolutions. For example, all algorithms are unable to detect the true partitions of the Karate network for λ=0.2, shown in **Fig. 5** (a). On the other hand, they obtain the average *NMI* equivalent to 1.0 for λ=0.3 except TJA_v-net. In addition, TJA-net displays a slight superiority over other three algorithms for increased value of λ whereas it performs much better than others for λ>0.8. As it is shown in **Fig. 5** (a), TJA-net and TJA_v-net are stable at most values of λ, especially for $0.4 \leq \lambda \leq 0.9$, and their *NMI* values just have slight fluctuations. Next, CSA-net slightly outperforms our algorithm on the dolphin network for $0.3<\lambda<0.6$, as shown in **Fig. 5** (b). Nonetheless, our algorithm shows a relative stability and superiority over other methods, by increasing value of λ especially for $\lambda > 0.7$. This is similar to the situation in **Fig. 5** (a). Finally, as for the football network, the *NMI* values in **Fig. 5** (c) illustrate that TJA-net can always maintain a high accuracy at different resolutions, where the results obtained by TJA_v-net also show a good performance. In specific, the average *NMI* values obtained by TJA-net are always approximately equal to 0.9 whereas Memetic-net and CSA-net resulting in varying values of *NMI* at different resolutions. On the other hand, the performance of CSA-net on the football network is very poor for $0.2 \leq \lambda \leq 0.6$. The comparison of the detecting results on these three real networks shows that the *NMI* values obtained by TJA-net and TJA_v-net have a relatively similar pattern. *NMI*s of Memetic-net and CSA-net also possess a similar pattern mainly because Memetic-net and CSA-net are based on the EA models whereas TJA-net and TJA_v-net are hybrid clustering methods, which intend to find an aggregation of nodes in the same community. In sum, TJA-net and TJA_v-net maintain a high detecting accuracy with strong stability at different resolutions.

Table 5 reports the average and the maximum *NMI* values on the structure known networks, i.e. N1, N2 and N3. Furthermore, **Table 6** reports the maximum *D* values on all the real-world networks. All algorithms run independently 30 times. '-' in **Table 6** indicates that such an algorithm could not obtain an effective result within the desired period of time. In **Table 5**, NMI_{max} represents the maximum value of *NMI* whereas NMI_{avg} represents the average value of *NMI* over 30 runs. In addition, the bold face denotes that TJA-net is not inferior to the ones obtained by other algorithms. MODPSO, CSA-net and our proposed algorithm, namely TJA-net, can correctly identify the real partitions of the karate and the dolphin network. In addition, our algorithm is slightly weaker than just MODPSO and Infomap in terms of NMI_{avg} on the football network whereas it is better than all other algorithms. Hence, by testing results of the three structure-known networks, we see that our algorithm is better than others. **Table 6** shows the maximum *D* values over 30 runs obtained by all algorithms. The results reported in **Table 6** show that our algorithm outperforms most other algorithms in terms of the *D* index for the small-scale networks. Nonetheless, just Memetic-net slightly outperforms our algorithm on the dolphin network and the football network. However, Memetic-net is unable to obtain results within a limited period of time for the large-scale networks whereas our proposed algorithm is able to get an approximate optimal solution within a reasonable time. Our algorithm outperforms other state-of-the-art algorithms in terms of the metric *D*.

In summary, the experimental results presented in this section show that our proposed algorithm has a distinct superiority over other algorithms in terms of the metrics *NMI* and *D*. Although the testing results by TJA-net on some small-scale networks, namely N2 and N3, are not the best, they are just slightly inferior to the best results obtained by Memetic-net. Furthermore, the value of the objective function *D* and of *NMI* are not consistent, i.e. their optimal value does not confirm one another. For example, the maximum value of *NMI* obtained by our algorithm on the dolphin network corresponds with non-optimal *D* value and it is slightly less than the maximum value of *D* obtained by Memetic-net.

In conclusion, we observe that using the objective function D_λ accompanies some limitations. Besides, the objective function determines, to a large extent, the merits and demerits of one algorithm and it is a difficult and urgent task to design a preeminent function for the networks with diverse structures.

3.5 The validity analysis of the proposed algorithm

In this section, we will confirm the validity of each strategy of the proposed algorithm, including (i) the preprocessing using a combination of LPA and KNN (ILPA), (ii) the community integration strategy and (iii) the refinement strategy will be verified.

3.5.1 Validity of the preprocessing strategy

In order to verify the advantages of the ILPA-based preprocessing (i.e. TJA-net) over using the LPA-based preprocessing (i.e. TJA_v-net), we conduct again a series of experiments on the LFR1 networks. We use experimental parameter settings identical to the ones used in section **3.4.1** and we consider λ being {0.2, 0.3, 0.4, ..., 1.0}. The maximum *NMI* values over 10 runs are plotted in **Fig.6**.

In **Fig.6,** we see that TJA-net always outperforms TJA_v-net. Even though the community structure is more ambiguous, i.e. the value of γ is greater than 0.50, TJA-net still maintain a high accuracy such that the corresponding *NMI* is not less than 0.7. However, the LPA-based technique, i.e. TJA_v-net, loses its efficiency, i.e. *NMI*=0.0, for the weak community structure. Therefore, we roughly conclude that the ILPA-based preprocessing strategy outperforms the LPA-based preprocessing strategy in network clustering.

3.5.2 Validity of community integration and refinement strategy

Here, we demonstrate the validity of the community integration strategy (CIS) and the refinement strategy (RS) and we obtain three comparison methods by combining our proposed components to verify them, namely ILPA, ILPA combined with the community integration strategy, i.e. ILPA+CIS and ILPA combined with the community integration strategy and the refinement strategy, i.e. TJA-net or ILPA+CIS+RS. All testing results are obtained at λ=0.3, and we only focus on the karate and the dolphin network for the sake of visualization, whose results are shown in **Figs.7** and **8**.

Fig.7 and **Fig.8** show that each step of TJA-net plays a very vital role in the overall framework and makes the detecting results more accurate. In addition, these figures show the ILPA-based preprocessing can effectively find out closely connected nodes and using CIS after that makes sub-communities merge into a large community. Finally,

we adopt RS so as to refine the performance achieved by the first two steps as it can be seen in the correction of node 10 in **Fig.7** and node 31 in **Fig.8**. Due to the good performance of the first two steps, there is only one node being misclassified in terms of the two small-scale cases where it is reclassified in the third stage. In summary, our algorithm is a powerful and efficient method of identifying partitions in a network, which consists of three effective components: (i) preprocessing, (ii) merging sub-communities and (iii) refinement of misclassified nodes.

3.6 The average running time of the algorithms

This experiment aims at comparing the running time of all algorithms. The average running time of each algorithm over 30 runs are reported in **Table 7**. All algorithms are applied on the same dataset. The boldface in this table represents that the corresponding algorithm has computation time larger than our proposed algorithm.

We used the code made available by the authors in order to ensure the impartiality of our experiments. This means we have to ignore the effect of different programming languages that the authors used on the running time. Hence, we adopt the same testing platform for all algorithms to minimize the interference of external factors. In **Table 7**, we observe that TJA-net and BGLL possess similar running time for networks with different scales. In addition, Memetic-net and CSA-net are not capable of dealing with large-scale networks. The running time of Infomap is very larger than the ones of MODPSO, BGLL, NM_CI-net and TJA-net for the large-scale networks, i.e. N4, N5 and N6. Nonetheless, because of the computation burden of the third stage of our proposed approach, which requires processing numerous boundary nodes, particularly, if the pending network holds many communities, the running times of MODPSO and NM_CI-net are less than ours for the large-scale networks, i.e. N5 and N6. Hence, the running time of TJA-net is larger than the ones of MODPSO and NM_CI-net for the network whose size is large. In spite of this, our proposed algorithm makes a very good trade-off between accuracy, stability and computation time.

4. Conclusion

In this paper, we proposed an algorithm to identify communities in a network. To deal with the networks at different resolutions, our proposed approach includes three effective stages (i) preprocessing and preliminary labelling, (ii) merging sub-communities and (iii) modifying the misclassified nodes. The preprocessing, called ILPA, assigns the same labels to the densely connected nodes forming some small clusters/sub-communities while we do not expect to have an exact clustering result at this stage. If the majority of the adjacent nodes to the node to be assigned a label, called pending node, do not have the same label, LPA cannot assign a correct label to that node because LPA uses only the labels of nodes adjacent to the pending node. Our proposed preprocessing stage resolves this shortcoming by considering both the label of adjacent nodes and their closeness degree to the pending node. This preprocessing strategy successfully deals with those pending networks possessing, overlapping nodes and vague structures. Furthermore, the first stage produces many sub-communities where they can be used to constitute a real bigger community. On the top of preprocessing, we propose a mutual community membership function

where the membership value of any two communities determines whether the two communities should be merged. This stage is a step to confirm the number of clusters, and TJA-net obtains a reasonable number of communities after using community integration strategy. There might be a few nodes that are wrongly clustered, particularly the boundary nodes, because of the simple preprocessing. To resolve this issue, a refinement strategy is performed at the last stage. To show the effectiveness of our proposed algorithm, we performed a series of experiments with the computer-generated networks as well as with some real-world networks. This provides us with networks with different resolutions and properties. The results on different networks illustrate the effectiveness of the three stages of the proposed algorithm. To validate our approach, we also apply several state-of-the-art algorithms to cluster the testing networks. Although those algorithms perform well, to some extent, our approach shows relative superiority in terms of accuracy and robustness.

In spite of the robustness and accuracy of our approach, we found via the experiments that the running time of TJA-net has yet to be optimized. On the other hand, although TJA-net outperforms most of the state-of-the-art algorithms mentioned in this paper in terms of the running time, it is inferior to MODPSO and NM_CI-net in this regard. For a large-scale network that the network contains a relatively large number of communities, our approach is demanded to deal with more boundary nodes. Furthermore, although the first two steps may result in a few misclassified nodes, TJA-net is required to perform the refinement computation on all the boundary nodes, which makes the computation time of the algorithm larger than MODPSO. In future works, we will work on reducing the needless searching for the boundary nodes that will greatly decrease the running time of TJA-net. Moreover, we will extend our work to deal with signed networks and we will focus on the issues of community detection on different types of networks.

Acknowledgment

We would like to express our sincere appreciation to the editors and the anonymous reviewers for their insightful comments, which have greatly helped us in improving the quality of the paper. This work was partially supported by the National Basic Research Program (973 Program) of China under Grant 2013CB329402, the National Natural Science Foundation of China, under Grants 61371201, the Program for Cheung Kong Scholars and Innovative Research Team in University under Grant IRT_15R53.

References

[1] J. Vlasblom, S.J. Wodak, Markov clustering versus affinity propagation for the partitioning of protein interaction graphs. BMC Bioinformatics 10 (1) (2009) 99.

[2] S. Wasserman, K. Faust, Social network analysis: Methods and applications, Contemporary Sociology 91 (435) (1994) 219-220.

[3] M.E.J. Newman, The structure of scientific collaboration networks, Proceedings of the National Academy of Sciences 98 (2) (2001) 404-409.

[4] R. Albert, H. Jeong, A.L. Barabási, Internet: Diameter of the world-wide web, Nature 401 (6749) (1999) 130-131.

[5] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (6684) (1998) 440-442.

[6] A.L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509-512.

[7] M. Girvan, M.E.J. Newman, Community structure in social and biological networks, Proceedings of the National Academy of Sciences 99 (12) (2002) 7821-7826.

[8] M.E.J. Newman, Networks: An Introduction, OUP Oxford 327 (8) (2010) 741–743.

[9] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, Physical Review E 69 (2) (2004) 026113.

[10] M.E.J. Newman, Spectral methods for community detection and graph partitioning, Physical Review E 88 (4) (2013) 042822.

[11] J.S. Wu, F. Wang, P. Xiang, Automatic network clustering via density-constrained optimization with grouping operator, Applied Soft Computing 38 (2016) 606-616.

[12] Q. Cai, L.J. Ma, M.G. Gong, A survey on network community detection based on evolutionary computation, International Journal of Bio-Inspired Computation 8 (2) (2016) 84-98.

[13] T.H. Ma, J.J. Zhou, M.L. Tang, Y. Tian, A. Al-Dhelaan, M. Al-Rodhaan, S. Lee, Social Network and Tag Sources Based Augmenting Collaborative Recommender System, IEICE Transactions on Information & Systems, E98-D (4) (2015) 902-910.

[14] U.N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, Physical Review E Statistical Nonlinear & Soft Matter Physics 76 (2) (2007) 036106.

[15] M. Rosvall, C.T. Bergstrom, Maps of random walks on complex networks reveal community structure, Proceedings of the National Academy of Sciences 105 (4) (2008) 1118-1123.

[16] V.D. Blondel, J.L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of Statistical Mechanics: Theory and Experiment 2008 (10) (2008) 155-168.

[17] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, D. Parisi, Defining and identifying communities in networks, Proceedings of the National Academy of Sciences of the United States of America 101 (9) (2004) 2658-2663.

[18] Y. Hu, H. Chen, P. Zhang, M. Li, Z. Di, Y. Fan, Comparative definition of community and corresponding identifying algorithm, Physical Review E 78 (2) (2008) 026121.

[19] S. Fortunato, D. Hric, Community detection in networks: A user guide, Physics Reports 659 (2016) 1-44.

[20] S. Fortunato, Community detection in graphs, Physics Reports 486 (3) (2010) 75-174.

[21] C. Pizzuti, Ga-net: A genetic algorithm for community detection in social networks//Parallel Problem Solving from Nature–PPSN X, Springer Berlin Heidelberg (2008) 1081-1090.

[22] M.G. Gong, B. Fu, L.C. Jiao, H.F. Du, Memetic algorithm for community detection in networks, Physical Review E 84 (5) (2011) 056101.

[23] C. Pizzuti, A multi-objective genetic algorithm to find communities in complex networks, IEEE Transactions on Evolutionary Computation 16 (3) (2012) 418-430.

[24] C. Shi, Z.Y. Yan, Y.N. Cai, B. Wu, Multi-objective community detection in complex networks, Applied Soft Computing 12 (2) (2012) 850-859.

[25] X.Z. Wen, L. Shao, Y. Xue, W. Fang, A rapid learning algorithm for vehicle classification, Information Sciences, 295 (1) (2015) 395-406.

[26] C.S. Yuan, X.M. Sun, R. Lv, Fingerprint Liveness Detection Based on Multi-Scale LPQ and PCA, China Communications, 13 (7) (2016) 60-65.

[27] R.H. Shang, J. Bai, L.J. Jiao, C. Jin, Community detection based on modularity and an improved genetic algorithm, Physica A Statistical Mechanics & Its Applications 392 (5) (2013) 1215–1231.

[28] M.G. Gong, Q. Cai, X.W. Chen, L.J. Ma, Complex Network Clustering by Multi-objective Discrete Particle Swarm Optimization Based on Decomposition, IEEE Transactions on Evolutionary Computation 18 (1) (2014) 82-97.

[29] X.D. Duan, C.R. Wang, X.D. Liu, Y.P. Lin, Web community detection model using particle swarm optimization, Computer Science 35 (3) (2008) 1074-1079.

[30] L.J. Ma, M.G. Gong, J. Liu, Q. Cai, L.J. Jiao, Multi-level learning based memetic algorithm for community detection, Applied Soft Computing 19 (2) (2014) 121–133.

[31] U. Brandes, D. Delling, M. Gaertler, R. Goerke, M. Hoefer, Z. Nikoloski, D. Wagner, Maximizing modularity is hard, ArXiv Preprint Physics/0608255 (2006).

[32] S. Fortunato, M. Barthelemy, Resolution limit in community detection, Proceedings of the National Academy of Sciences 104 (1) (2007) 36-41.

[33] J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth, Knowledge and Information Systems 42 (1) (2015) 181-213.

[34] Z.P. Li, S.H. Zhang, R.S. Wang, X.S. Zhang, L.N. Chen, Quantitative function for community detection, Physical review E 77 (3) (2008) 036109.

[35] C.H. Mu, J. Xie, Y. Liu, F. Chen, Y. Liu, L.C. Jiao, Memetic algorithm with simulated annealing strategy and tightness greedy optimization for community detection in networks, Applied Soft Computing 34 (C) (2015) 485-501.

[36] Q. Cai, M.G. Gong, L.J. Ma, L.C. Jiao, A novel clonal selection algorithm for community detection in complex networks, Computational Intelligence 31 (3) (2015) 442-464.

[37] R.H. Shang, W.T. Zhang, L.C. Jiao, R. Stolkin, Y. Xue, A community integration strategy based on an improved modularity density increment for large-scale networks, Physica A: Statistical Mechanics and its Applications 469 (2017) 471-485.

[38] J.A. Hartigan, M.A. Wong, A K-means clustering algorithm, Applied Statistics, 28 (1) (1979) 100-108.

[39] J.C. Bezdek, R. Ehrlich, W. Full, FCM: The fuzzy c-means clustering algorithm, Computers & Geosciences 10 (2-3) (1984) 191-203.

[40] J. Zhang, Q. Tang, P. Li, D. Deng, Y Chen, A modified MOEA/D approach to the solution of multi-objective optimal power flow problem, Applied Soft Computing 47 (C) (2016) 494-514.

[41] J. Zhang, Q. Tang, Y. Chen, S. Lin, A hybrid particle swarm optimization with small population size to solve the optimal short-term hydro-thermal unit commitment problem, Energy 109 (2016) 765-780.

[42] Y.H. Zhang, X.M. Sun, B.W. Wang, Efficient Algorithm for K-Barrier Coverage Based on Integer Linear Programming, China Communications, 13 (7) (2016) 16-23.

[43] J. Zhang, Y. Wu, Y. Guo, B. Wang, H. Wang, H. Liu, A hybrid harmony search algorithm with differential evolution for day-ahead scheduling problem of a microgrid with consideration of power flow constraints, Applied Energy, 183 (2016) 791-804.

[44] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory 13 (1) (1967) 21-27.

[45] L.E. Peterson, K-nearest neighbor, Scholarpedia 4 (2) (2009) 1883.

[46] R.H. Shang, S. Luo, Y.Y. Li, L.C. Jiao, R. Stolkin, Large-scale community detection based on node membership grade and sub-communities integration, Physica A: Statistical Mechanics and its Applications 428 (2015) 279-294.

[47] M. Tasgin, A. Herdagdelen, H. Bingol, Community Detection in Complex Networks Using Genetic Algorithms, Eprint Arxiv 2005 (3120) (2006) 1067-1068.

[48] J. Handl, J. Knowles, An evolutionary approach to multiobjective clustering, IEEE transactions on Evolutionary Computation 11 (1) (2007) 56-76.

[49] M.J. Barber, J.W. Clark, Detecting network communities by propagating labels under constraints, Physical Review E Statistical Nonlinear & Soft Matter Physics 80 (2) (2009) 026129.

[50] J. Eustace, X. Wang, Y. Cui, Community detection using local neighborhood in complex networks, Physica A: Statistical Mechanics and its Applications 436 (2015) 665-677.

[51] L. Angelini, S. Boccaletti, D. Marinazzo, M. Pellicoro, S. Stramaglia, Identification of network modules by optimization of ratio association, Chaos: An Interdisciplinary Journal of Nonlinear Science 17 (2) (2007) 023114.

[52] Y.C. Wei, C.K. Cheng, Ratio cut partitioning for hierarchical designs, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 10 (7) (1991) 911-921.

[53] F. Wu, B.A. Huberman, Finding communities in linear time: a physics approach, The European Physical Journal B-Condensed Matter and Complex Systems 38 (2) (2004) 331-338.

[54] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Physical Review E Statistical Nonlinear & Soft Matter Physics 78 (4 Pt 2) (2008) 046110.

[55] A. Lancichinetti, S. Fortunato, Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities, Physical Review E Statistical Nonlinear & Soft Matter Physics 80 (1 Pt 2) (2009) 016118.

[56] W.W. Zachary, An Information Flow Model for Conflict and Fission in Small Groups, Journal of Anthropological Research 33 (4) (1977) 452-473.

[57] D. Lusseau, K. Schneider, O.J. Boisseau, P. Haase, E. Slooten, S.M. Dawson, The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations, Behavioral Ecology and Sociobiology 54 (4) (2003) 396-405.

[58] M.E.J. Newman, Finding community structure in networks using the eigenvectors of matrices, Physical Review E 74 (3) (2006) 036104.

[59] M. Boguñá, R. Pastor-Satorras, A. Díaz-Guilera, A. Arenas, Models of social networks based on social distance attachment, Physical Review E 70 (5) (2004) 056122.

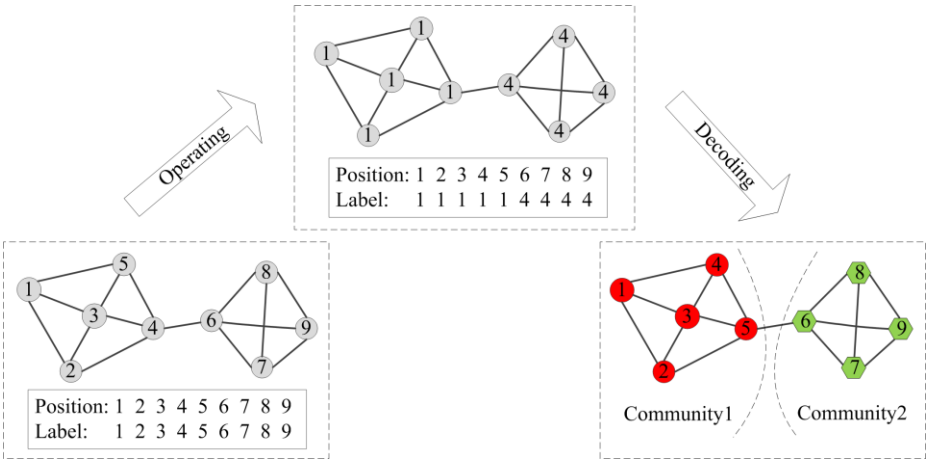
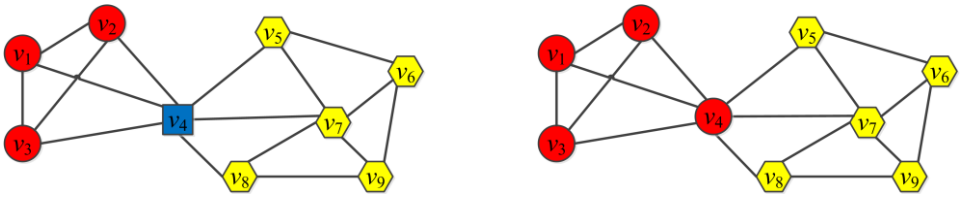


Fig.1. The representation and decoding of the proposed algorithm



(a) (b)

Fig.2. (a) A case that cannot be handled by LPA. (b) The same situation that can be correctly handled by ILPA.

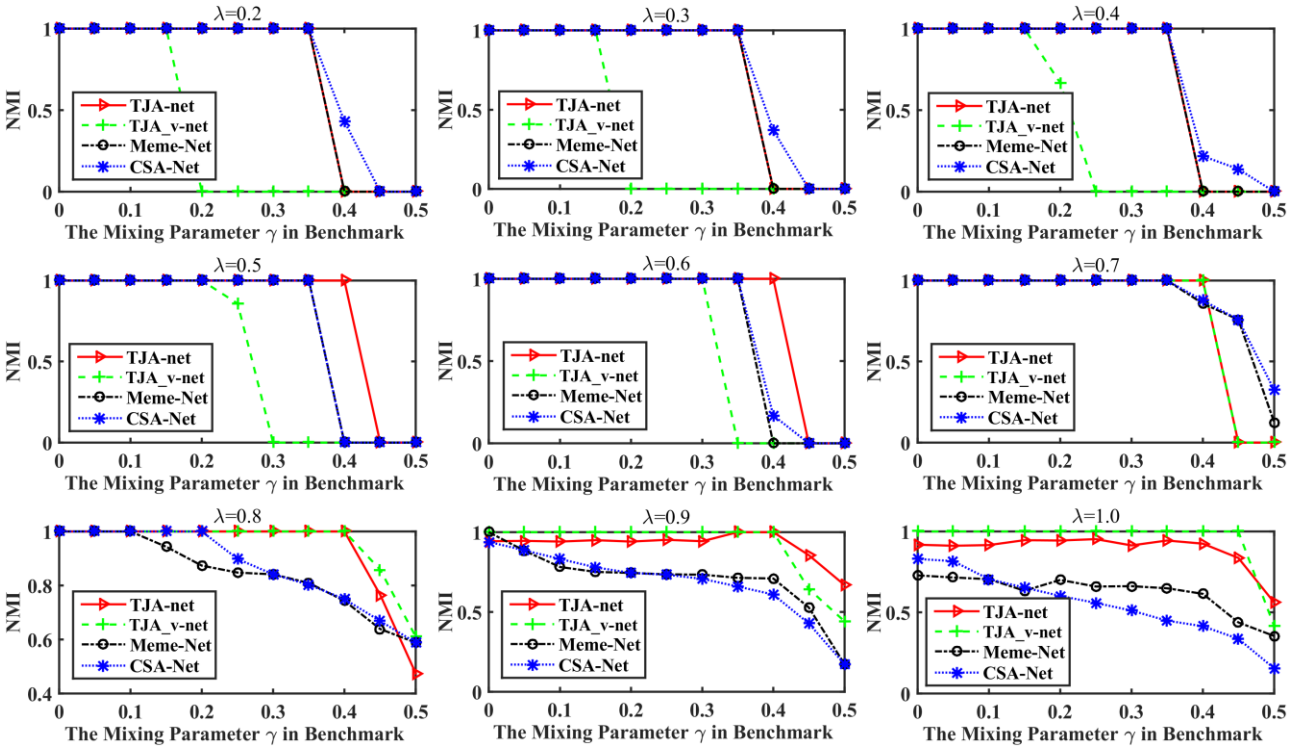
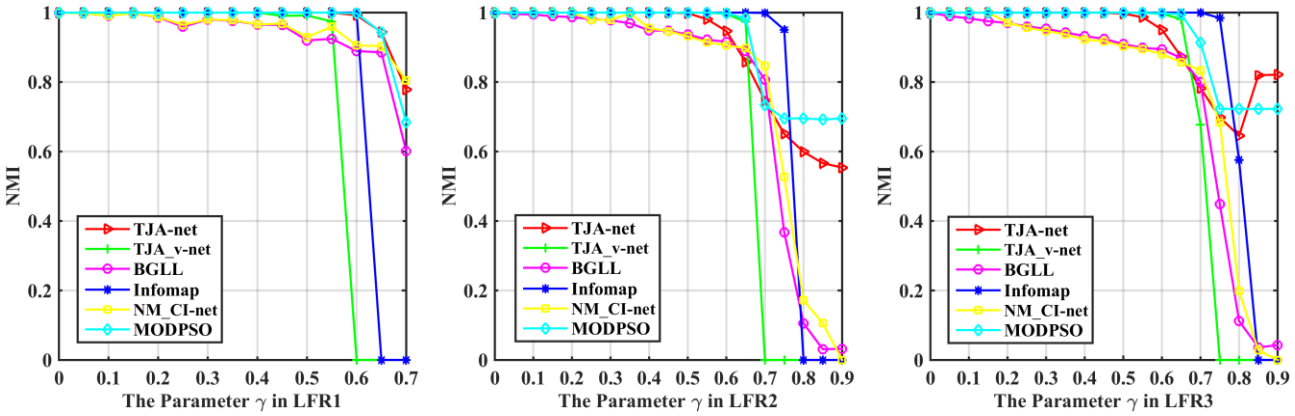
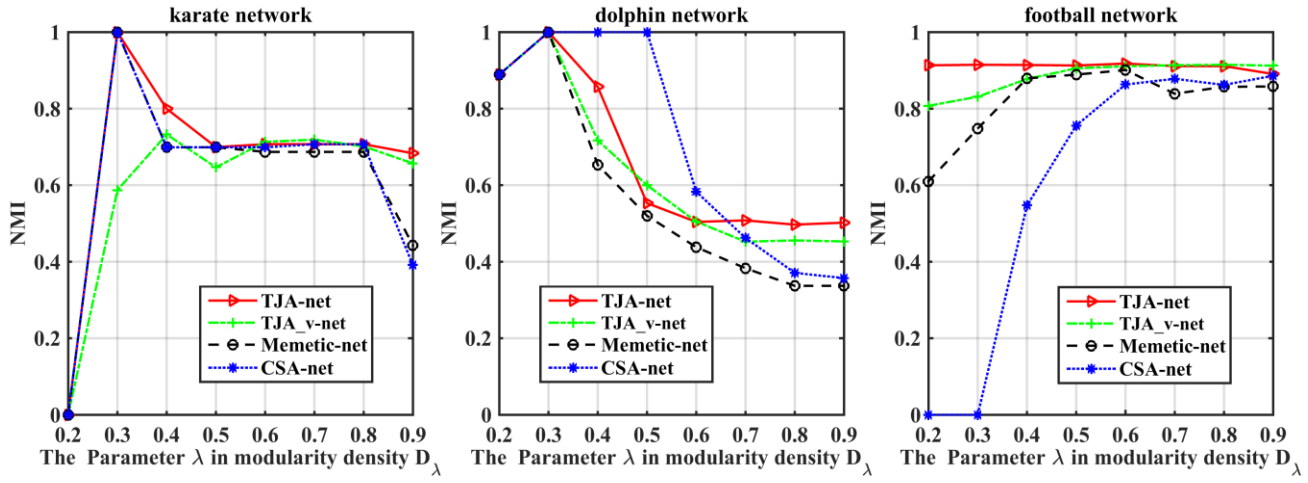


Fig.3. Maximum *NMI* values on GN extended benchmark networks



(a) (b) (c)

Fig.4. Maximum *NMI* values on LFR1, LFR2 and LFR3. (a) Maximum *NMI* values on LFR1. (b) Maximum *NMI* values on LFR2. (c) Maximum *NMI* values on LFR3.



(a) (b) (c)

Fig. 5. Average *NMI* values at different values of λ. (a) Experimental results on N1. (b) Experimental results on N2. (c) Experimental results on N3.

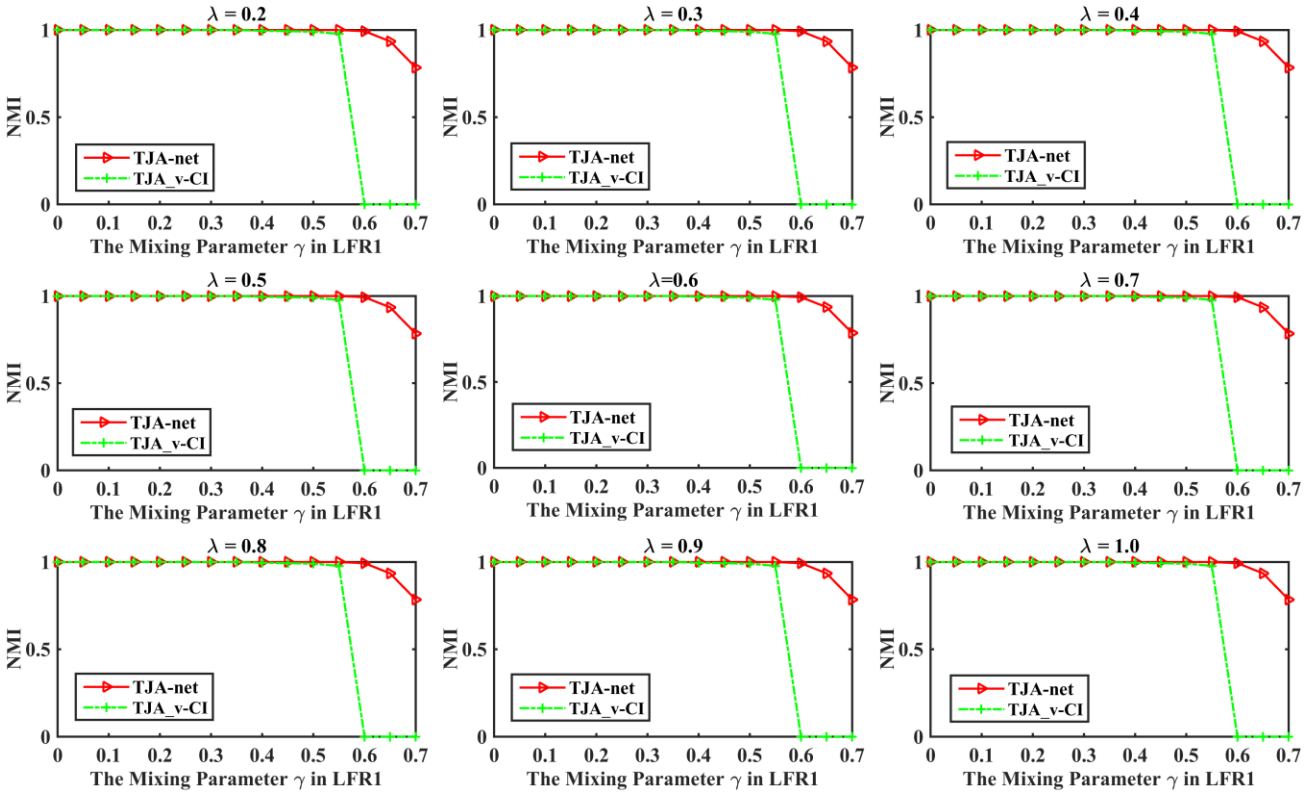
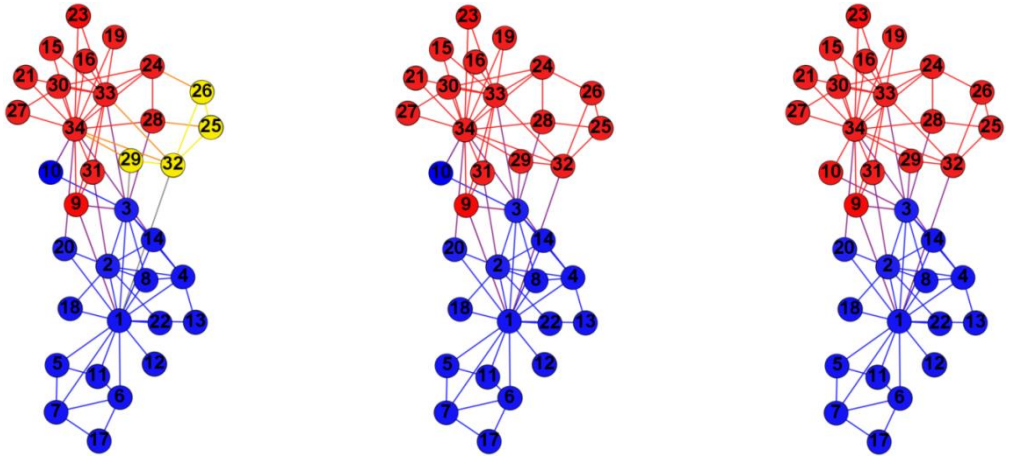
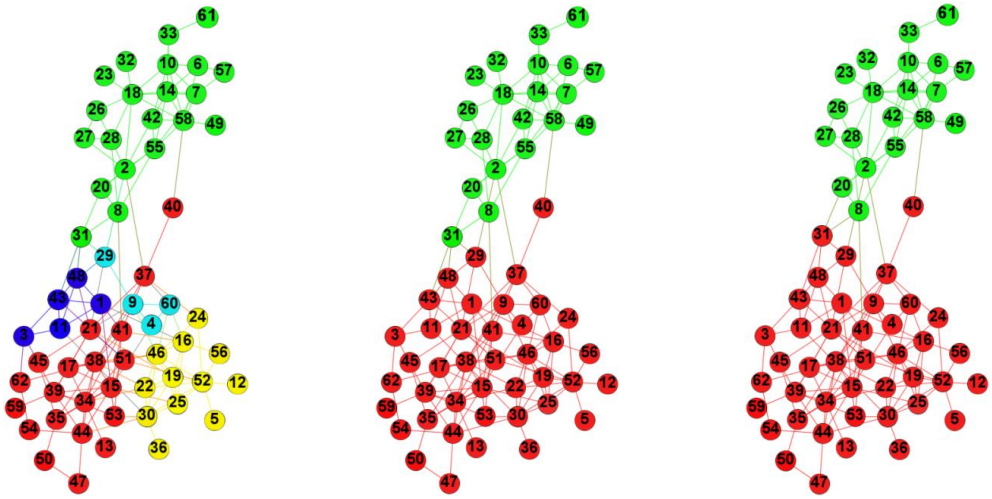


Fig.6. Maximum *NMI* values on LFR1 by TJA-net and TJA_v-net



(a) (b) (c)

Fig.7. The detected communities on the karate network. (a) The result by using ILPA, *NMI*=0.699. (b) The result by using ILPA+CIS, *NMI*=0.837. (c) The result by using ILPA+CIS+RS, *NMI*=1.000.



(a) (b) (c)

Fig.8. The detected communities on the dolphin network. (a) The result by using ILPA, *NMI*=0.558. (b) The result by using ILPA+CIS, *NMI*=0.889. (c) The result by using ILPA+CIS+RS, *NMI*=1.000.

Table 1. The closeness values between v_4 and its adjacent nodes

Adjacent nodes	v_1	v_2	v_3	v_5	v_7	v_8
Community label	'1'	'1'	'1'	'2'	'2'	'2'
Closeness	3	3	3	2	3	2

Table 2. The information of the real-world networks

Network	Node number	Edge number	Average degree	Real clusters	Reference
Karate(N1)	34	78	4.59	2	[56]
Dolphin(N2)	62	159	5.13	2	[57]
Football(N3)	115	613	10.66	12	[7]
Net-science(N4)	1589	2742	3.45	Unknown	[58]
Power grid(N5)	4941	6594	2.67	Unknown	[5]
PGP(N6)	10680	24340	4.55	Unknown	[59]

Table 3. Experimental results on GN extended benchmark networks at the mixing parameter λ={0.3, 0.6, 0.9}

The mixing parameter γ		0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
Memetic-net	λ=0.3	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000
	λ=0.6	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000
	λ=0.9	0.782	0.751	0.744	0.735	0.733	0.713	0.709	0.528	0.172
CSA-net	λ=0.3	1.000	1.000	1.000	1.000	1.000	1.000	0.369	0.000	0.000
	λ=0.6	1.000	1.000	1.000	1.000	1.000	1.000	0.169	0.000	0.000
	λ=0.9	0.832	0.779	0.744	0.734	0.708	0.659	0.609	0.428	0.172
TJA_v-net	λ=0.3	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	λ=0.6	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000
	λ=0.9	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.644	0.440
TJA-net	λ=0.3	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.000	**0.000**	**0.000**
	λ=0.6	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.000**	**0.000**
	λ=0.9	0.942	0.949	0.942	0.952	0.944	**1.000**	**1.000**	**0.856**	**0.667**

Table 4. Experimental results on the LFR3 networks at $\lambda = 0.5$

Algorithms		BGLL	Infomap	NM_CI-net	MODPSO	TJA_v-net	TJA-net
LFR3	γ=0.00	0.998	1.000	1.000	1.000	1.000	**1.000**
	γ=0.05	0.989	1.000	1.000	0.999	1.000	**1.000**
	γ=0.10	0.983	1.000	1.000	0.999	1.000	**1.000**
	γ=0.15	0.975	1.000	0.999	0.999	1.000	**1.000**
	γ=0.20	0.969	1.000	0.970	0.999	1.000	**1.000**
	γ=0.25	0.959	1.000	0.956	0.999	1.000	**1.000**
	γ=0.30	0.953	1.000	0.946	0.999	1.000	**1.000**
	γ=0.35	0.942	1.000	0.937	0.999	1.000	**1.000**
	γ=0.40	0.932	1.000	0.921	0.999	1.000	**1.000**
	γ=0.45	0.924	1.000	0.917	0.999	1.000	0.999
	γ=0.50	0.909	1.000	0.902	0.999	0.999	0.996
	γ=0.55	0.898	1.000	0.894	0.999	0.999	0.987
	γ=0.60	0.894	1.000	0.880	0.997	0.996	0.951
	γ=0.65	0.868	0.999	0.856	0.995	0.980	0.873
	γ=0.70	0.800	0.984	0.833	0.913	0.678	0.782
	γ=0.75	0.448	0.575	0.683	0.722	0.000	0.698
	γ=0.80	0.112	0	0.197	0.722	0.000	0.645
	γ=0.85	0.036	0	0.027	0.722	0.000	**0.819**
	γ=0.90	0.042	0	0.000	0.722	0.000	**0.821**

Table 5. *NMI* values on three networks with known ground truths

Network	Index	Memetic-net	MODPSO	Infomap	CSA-net	BGLL	NM_CI-net	TJA-net
N1	NMI_{max}	1	1	0.700	1	0.587	0.700	**1**
	NMI_{avg}	0.860	1	0.700	1	0.587	0.649	**1**
N2	NMI_{max}	1	1	0.562	1	0.516	0.638	**1**
	NMI_{avg}	0.785	1	0.562	1	0.516	0.586	**1**
N3	NMI_{max}	0.862	0.927	0.924	0.886	0.890	0.911	**0.927**
	NMI_{avg}	0.774	0.926	0.924	0.861	0.890	0.879	0.915

Table 6. Maximum *D* values on six real-world networks

Network	Index	Memetic-Net	MODPSO	Infomap	CSA-net	BGLL	NM_CI-net	TJA-net
N1	D_{max}	7.845	7.842	7.845	7.845	7.464	7.845	**7.845**
N2	D_{max}	11.707	10.810	10.076	10.837	10.208	11.017	11.421
N3	D_{max}	44.340	40.165	42.846	29.321	44.142	42.2213	44.190
N4	D_{max}	—	727.820	728.716	689.925	605.676	629.9433	**749.284**
N5	D_{max}	—	664.231	691.937	—	57.306	99.7342	**789.363**
N6	D_{max}	—	1032.930	1721.021	—	262.673	257.3625	**1929.560**

Table 7. The average running time of the algorithms (Unit：Sec/each)

Network	Memetic-net	MODPSO	Infomap	CSA-net	BGLL	NM_CI-net	TJA-net
N1	**1.5121**	**0.1981**	**0.0810**	**0.3032**	**0.0179**	**0.5802**	0.0031
N2	**5.9804**	**0.3307**	**0.0940**	**0.5413**	**0.0332**	**0.7917**	0.0078
N3	**25.3414**	**0.6927**	**0.7660**	**1.0702**	**0.0477**	**1.1679**	0.0317
N4	**—**	**12.3230**	**50.591**	**32.8663**	**11.8214**	**8.7645**	4.3866
N5	**—**	88.195	**6748.140**	|	**310.894**	50.306	301.984
N6	**—**	489.44	**17033.90**	|	1215.60	138.53	1368.34